**Python Random Module Reference**

Python has a built-in module that you can use to make random numbers.

The random module has a set of methods:

|  |  |
| --- | --- |
| **Method** | **Description** |
| [seed()](https://www.w3schools.com/python/ref_random_seed.asp) | Initialize the random number generator |
| [getstate()](https://www.w3schools.com/python/ref_random_getstate.asp) | Returns the current internal state of the random number generator |
| [setstate()](https://www.w3schools.com/python/ref_random_setstate.asp) | Restores the internal state of the random number generator |
| [getrandbits()](https://www.w3schools.com/python/ref_random_getrandbits.asp) | Returns a number representing the random bits |
| [randrange()](https://www.w3schools.com/python/ref_random_randrange.asp) | Returns a random number between the given range |
| [randint()](https://www.w3schools.com/python/ref_random_randint.asp) | Returns a random number between the given range |
| [choice()](https://www.w3schools.com/python/ref_random_choice.asp) | Returns a random element from the given sequence |
| [choices()](https://www.w3schools.com/python/ref_random_choices.asp) | Returns a list with a random selection from the given sequence |
| [shuffle()](https://www.w3schools.com/python/ref_random_shuffle.asp) | Takes a sequence and returns the sequence in a random order |
| [sample()](https://www.w3schools.com/python/ref_random_sample.asp) | Returns a given sample of a sequence |
| [random()](https://www.w3schools.com/python/ref_random_random.asp) | Returns a random float number between 0 and 1 |
| [uniform()](https://www.w3schools.com/python/ref_random_uniform.asp) | Returns a random float number between two given parameters |
| [triangular()](https://www.w3schools.com/python/ref_random_triangular.asp) | Returns a random float number between two given parameters, you can also set a mode parameter to specify the midpoint between the two other parameters |
| betavariate() | Returns a random float number between 0 and 1 based on the Beta distribution (used in statistics) |
| expovariate() | Returns a random float number based on the Exponential distribution (used in statistics) |
| gammavariate() | Returns a random float number based on the Gamma distribution (used in statistics) |
| gauss() | Returns a random float number based on the Gaussian distribution (used in probability theories) |
| lognormvariate() | Returns a random float number based on a log-normal distribution (used in probability theories) |
| normalvariate() | Returns a random float number based on the normal distribution (used in probability theories) |
| vonmisesvariate() | Returns a random float number based on the von Mises distribution (used in directional statistics) |
| paretovariate() | Returns a random float number based on the Pareto distribution (used in probability theories) |
| weibullvariate() | Returns a random float number based on the Weibull distribution (used in statistics) |

***1. Python Random seed() Method***

*Example*

Set the seed value to 10 and see what happens:

import random  
random.seed(10)  
print(random.random())

# The generator creates a random number based on the seed value, so if the seed value is 10, you will always get 0.5714025946899135 as the first random number.

*Output:* 0.5714025946899135

**Definition and Usage**

The seed() method is used to initialize the random number generator.

The random number generator needs a number to start with (a seed value), to be able to generate a random number.

By default the random number generator uses the **current system time**.

Use the seed() method to customize the start number of the random number generator.

**Note:** If you use the same seed value twice you will get the same random number twice. See example below

**Syntax**

random.seed(*a*, *version*)

**Parameter Values**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *a* | Optional. The seed value needed to generate a random number. If it is an integer it is used directly, if not it has to be converted into an integer. Default value is None, and if None, the generator uses the current system time. |
| *version* | An integer specifying how to convert the a parameter into a integer. Default value is 2 |

**More Examples**

*Example*

Demonstrate that if you use the same seed value twice, you will get the same random number twice:

import random  
random.seed(10)  
print(random.random())  
  
random.seed(10)  
print(random.random())

*Output:*

0.5714025946899135  
0.5714025946899135

Python's random number generator relies on a deterministic algorithm to produce a sequence of pseudo-random numbers based on an initial seed value. The seed acts as a starting point for the algorithm, ensuring that the sequence of generated numbers can be reproduced if the same seed is used again. This feature is particularly useful for debugging, simulations, and scenarios where reproducibility is essential.

When the random.seed() function is called, it initializes the internal state of the random number generator with the provided seed value. If no seed is specified, Python uses the current system time or other sources of entropy to generate a seed automatically. The seed value influences the sequence of numbers generated by subsequent calls to functions like random.random(), random.randint(), and others.

## **How the seed value is used to generate random numbers**

The core of Python's random number generation is based on a pseudorandom number generator (PRNG), specifically the Mersenne Twister algorithm. When a seed is set, the PRNG initializes its internal state with that seed. From this state, it produces a sequence of numbers that appear random but are entirely determined by the seed. This deterministic process ensures that given the same seed, the sequence of random numbers will be identical across different runs.

## **Role of the second parameter in the seed function**

## The random.seed() function can accept various types of seed values, including integers, floating-point numbers, strings, bytes, and even None. When a non-integer seed is provided, Python internally converts it into an integer to initialize the PRNG. The second parameter, often referred to as version in some contexts, determines how the seed value is processed, especially when the seed is not an integer.

In Python's implementation, if the seed value is not an integer, the function attempts to convert it into an integer using a hashing or encoding process. For example, if a string or bytes are provided, Python computes a hash value or uses a specific encoding method to produce an integer seed. This conversion ensures that the seed is always an integer, which is necessary for the internal workings of the PRNG.

In summary, the seed value initializes the random number generator's internal state, and the second parameter influences how non-integer seed values are converted into integers. This process guarantees that the seed can be derived from various data types, maintaining the reproducibility and flexibility of Python's random number generation.